XML 技术在软件测试框架中的应用

胥光辉[1]　杜　瑜[1*]　李　翔[2]　徐永森[3]
(1. 解放军理工大学指挥自动化学院计算机系, 江苏 南京　210007; 2. 华东工程软件测评中心, 上海　200233;
3. 南京大学计算机科学与技术系, 江苏 南京　210093)

摘　要: 对国内外软件测试工具的系统框架进行研究,在此基础上提出一种扩展性更好的系统框架设计方案。在本文提出的系统框架中,源程序信息使用 XML 格式的文件表示,根据具体应用,系统使用不同的信息提取模板文件(XSLT 格式)对应用关心的程序信息进行提取。随着应用的增加,系统只需要扩充相应的信息提取模板文件,而不用改变源程序信息文件。

关键词: 软件测试; 系统框架; XML; XSLT
中图分类号: TP31

引　言

随着软件技术的迅速发展,软件系统的规模和复杂度日益增大。很多应用领域对软件的可靠性要求甚高。软件测试作为保证软件质量最有效的方法,受到广泛关注。软件测试工具作为软件测试的重要辅助手段,近年来成为研究的热点。目前,市场上主流的软件测试工具是部分欧美厂商的产品,其中典型代表有 Telelogic 公司的 Logiscope, AMC 公司的 CodeTEST, McCabe 公司的 McCabe 工具等。国内较成熟的商用产品是由北京航空航天大学软件工程研究所研发的四大软件测试工具: QESat/Java、QESuite、EPMS 和 UML_Designer。南京大学计算机软件新技术国家重点实验室对软件测试技术进行研究,研发了一套针对 C/C++ 语言的嵌入式软件测试工具集 ——EASTT,不过该工具集离商用化产品还有一段距离。

本文对上述这些工具的系统框架设计进行了深入的研究,发现了其在扩展性方面的不足,以此为基础提出了一种基于 XML 的软件测试系统框架的解决方案,并实现一个原型系统,验证了方案的有效性。

收稿日期: 2007-05-21
基金项目: 国家自然科学基金(60303023)
第一作者: 男,1970 年生,副教授,博士
*通讯联系人
E-mail: dudupla@163.com

1　系统框架

良好的软件测试系统框架不仅可以提高软件测试工具的测试效率,还能保证软件测试的正确性[1-4]。主流软件测试工具一般采用图 1 所示的框架结构。

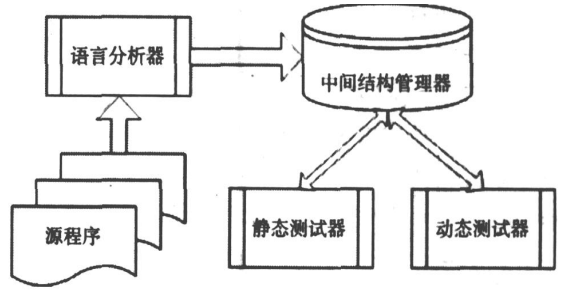

图 1　软件测试工具系统框架

Fig. 1　System framework of software test tool

1.1　语言分析器(LGA)

LGA 是软件测试工具中直接处理源程序的模块,负责对软件源代码进行扫描、分析,产生分析结果,并将分析结果提供给中间结构管理器。LGA 完成词法分析、语法分析和部分语义分析工作。

LGA 作为软件测试工具的前端,与之交互的内部模块主要是中间结构管理器(ISM)。LGA 在分析的过程中调用 LGA 与 ISM 间的接口,将分析结果动态地传递给 ISM,在语法分析结束后,再将其他模块需要的信息一次传递给 ISM。

1.2　中间结构管理器(ISM)[5]

ISM 是软件测试工具的核心组成部分。ISM 负责构建和维护软件测试过程中的核心数据结构,

并为其他功能模块提供原始数据。

ISM 的另一个重要功能是对被测程序进行分析,对基本测试成分进行划分,以及完成软件质量标准中的基本 Metrics 计算。

ISM 的结构如图 2 所示。

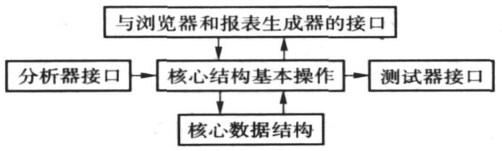

图 2 ISM 结构图

Fig. 2 ISM structure

1.3 动态测试器(DYT)[6-9]

DYT 是一组 CAST(Computer Aided Software Testing)工具,帮助用户设计、运行和评估整个测试过程,其基本测试方法是结构性测试,以测试覆盖率作为评价测试充分性的重要手段。

DYT 从 ISM 获取源程序的相关信息,经过处理,向界面模块提供包括覆盖率、覆盖频度、测试效率、测试实例集合最小化等信息。

1.4 静态分析器(STA)[10]

STA 由静态分析工具界面、结果浏览工具界面、测试分析工具界面、文档生成器界面、软件度量标准编辑器界面 5 个部分组成。

STA 在 ISM 提供的源程序信息的基础上进行静态分析、程序结构分析和软件度量等操作,并将分析和度量结果以表格、直方图、饼图和 kiavia 图等合适的方式展示出来,供用户浏览。

2 系统改进

2.1 原有系统框架的不足

从本文第一部分的简介中可以看到,动态测试器和静态分析器中包含的功能作为软件测试工具系统的后端应用,对源程序中不同信息的关注程度不尽相同。经过分析,发现该系统框架在其核心组成模块 ISM 的设计中,存在以下两点不足之处。

(1) ISM 从语言分析器获取源程序的信息并构造核心数据结构。根据系统后端应用需求,在语言分析器中需要将应用所需的信息全部分析出来。这样一来,势必会造成如下一种局面:假如增加新的应用,而满足该应用的源程序信息不存在或部分存在于现有的核心数据结构中,我们不得不修改语言分析器,使其提供给 ISM 模块的分析结果中包含新增加应用需要的信息。由于系统后端应用的不断扩充,对语言分析器也需要进行反复的修改,以致影响测试工具的稳定性,同时也造成了测试工具维护成本的增加。

(2) ISM 管理和维护的信息量过于庞大。ISM 从 LGA 获得源程序的所有词法、语法分析信息,构建和维护核心数据结构。该核心数据结构势必包含海量的源程序信息,系统后端应用(如:静态分析、动态测试、结果显示等)需要根据具体应用的要求,从核心数据结构中获取相关的信息。从海量的源程序信息中提取少量相关信息,这项工作较为繁重,效率也颇低,并且,对于不同的应用,提取的信息也不尽相同,反复地做类似的工作,会明显地造成资源浪费。当对于某一应用需要进行反复测试时,更会造成反复的、不必要的信息提取。

2.2 解决方案

本文从上述两个主要问题出发,提出如下解决方案。

(1) 使系统后端的应用与语言分析器分离。当后端应用增加时,现有的语言分析器的分析结果仍然能够满足新增应用需求,避免对语言分析器进行反复修改,同时也避免了在源程序不改变的前提下,对源程序进行反复分析。

(2) 语言分析器将分析结果一次生成源信息文件(XML 文件),该源信息文件包含源程序的所有信息,在源程序不改变的前提下,根据应用的扩充,只需要不断扩充信息提取模板文件(XSLT 文件),不必对源程序进行重复的分析。源信息文件经过信息提取模板文件处理,生成应用信息文件(XML 文件)。针对不同的应用,使用与应用相关的信息提取模板文件就能生成不同的应用信息文件,即使某一应用需要反复测试,也无需再对海量的源信息文件进行处理,只需要处理相应的应用信息文件即可。

根据上述思想,改进后的软件测试工具系统框架如图 3 所示。

改进后的 ISM 由源信息文件、信息提取模板文件和应用信息文件 3 部分组成。源信息文件包含海量的源程序信息;信息提取模板文件可根据具体应用进行扩充;应用信息文件是源信息文件经过信息提取模板文件处理后,生成的与应用相关的信息文件,该文件包含的信息只是源信息文件中的一部分信息(满足具体应用所需要的信息部分)。

中间结构管理器前端的语言分析器,可以采用

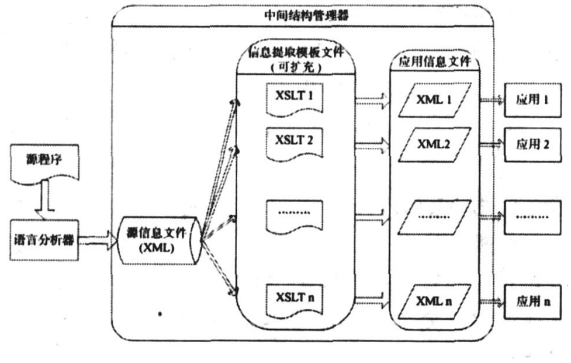

图 3　改进后的软件测试工具系统框架

Fig. 3　Improved system framework of software test tool

标准的语言分析工具进行,使得源信息文件中保存的源程序信息尽可能丰富。

3　系统实现

根据 2.2 节所述的解决方案,基于 Java 语言,本文给出一个简单的原型系统实现。

3.1　语言分析器

目前,市场上的语言分析工具数不胜数,可以根据软件使用成本、分析效率、分析结果优劣等综合条件进行选择,同时也可以自行开发语言分析工具。本文给出的原型系统中,语言分析工具使用的是 Nor Ken 公司开发的分析引擎 Pro Grammar。该分析引擎,根据被分析程序语言的文法规则定义对本分析程序进行分析,形成一棵解析树。

本文实现的原型系统是基于 Java 语言进行分析测试,因此,文法规则根据 Java 语言规范进行定义。

3.2　源信息文件生成

Pro Grammar 根据文法规则定义,对被测试程序进行分析,得到引擎自定义的分析结果文件(非 XML 文件)。使用 Pro Grammar 提供的 API,将分析结果文件转换成我们需要的 XML 格式的文件。

Pro Grammar 分析结果文件转换为 XML 文件过程为

源程序代码:

```
1  import java.io. *;
2  public class Test{
3    public static void main(){
4      int count = 4;
5      if(count < = 4) {
6        System.out.println("success count !");
7      } else {
8        System.out.println("error count !");
9      }
10    for(int i = 0; i < count; i + +) {
11      System.out.println("i:" + i);
12   }
13  }
14 }
```

源程序经过语言分析器分析处理,得到源信息文件,代码为

```
1  - <compilation_unit >
2    + <import_statement >
3    - <type_declaration >
4      - <class_declaration >
5        + <modifiers >
6          <class_name > Test </class_name >
7          - <field_declarations >
8            - <field-declaration >
9              + <method-declaration >
10             + <modifiers >
11             + <type >
12               <method_name > main </method_name >
13             - <statement_block >
14               - <statement >
15                 + <variable_declaration >
16                 </statement >
17               - <statement >
18                 + <if_statement >
19                 </statement >
20               - <statement >
21                 + <for_statement >
22                 </statement >
23             </statement_block >
24           </method_declaration >
25         </field_declaration >
26       </field_declarations >
27     </class_declaration >
28   </type_declaration >
29 </compilation_unit >
```

从上述过程中可以看到,源程序中第 1 行"import 语句"经过分析、转换后对应于源 XML 文件中的第 2 行"import_statement"标签;第 4 行定义变量语句对应于源 XML 文件中的第 15 行"variable_

declaration”标签;第 5～9 行“if 语句”对应源 XML 文件中的第 18 行“if_statement”标签;第 10～12 行“for 语句”对应源 XML 文件中的第 21 行“for_statement”标签。通过上述分析,可以看到,源程序在经过分析、转换后,其包含的原有信息都能不丢失地保存在源 XML 文件中,保证了源程序信息的完整性。

3.3 信息提取

源信息文件中包含几乎所有的源程序信息,针对具体的应用,只需要部分的源程序信息,因此,需要对源信息文件进行筛选,提取与应用相关的信息部分。本文针对“分支覆盖”应用实现原型系统,只需要提取源程序中 if 分支语句部分的信息(即源信息文件中第 17～19 行之间的部分)。

根据上述要求,定义信息提取模板文件(XSLT 文件)如下所示。

```
1  <?xml version="1.0"encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0"xmlns:xsl="
     http://www.w3.org/1999/XSL/Transform"
     >
3  <xsl:output method="xml"version="1.0"en-
     coding="UTF-8"indent="yes">
4    <xsl:template match="/">
5      <branchcoverage>
6        <xsl:for-each
         select="//statement/if_statement">
7          <statement>
8            <xsl:copy-of select="."/>
9          </statement>
10       </xsl:for-each>
11       </branchcoverage>
12   </xsl:template>
13 </xsl:stylesheet>
```

信息提取模板文件遵循 XML 格式进行定义,下面对文件中主要的代码进行解释。在本文给出的信息提取模板文件中,第 4～12 行是代码的主要部分。第 4 行代码中的“”/””代表源 XML 文件中根节点“compilation_unit”的父节点,该父节点是一个虚根节点,实际并不存在,仅仅定义对源 XML 文件进行搜索的起始位置;第 5 行定义应用信息文件中的根节点;第 6 行使用“xsl:for_each”函数对源 XML 文件中所有满足属性“select”的节点进行处理,属性“select”表示从虚根节点开始搜索“statement”节点,若“statement”节点的子节点是“if_statement”节点则满足条件;第 7 行定义应用信息文件中的节点;第 8 行使用“xsl:copy-of”函数使用属性“select=“.””对“if_statement”及其所有后续节点进行复制。

使用 ALTOVA 公司的 XMLSpy 软件,根据信息提取模板文件的定义,对源 XML 文件中 if 分支语句对应的信息进行匹配,匹配成功后 XMLSpy 会自动提取相关信息。经过匹配提取后,可以得到应用信息文件,如下所示。

```
1  <?xml version="1.0"encoding="UTF-8"?
   >
2  - <branchcoverage>
3    - <statement>
4      - <if_statement>
5          + <expression>
6          + <statement>
7          + <statement>
8        </if_statement>
9    </statement>
10 </branchcoverage>
```

对上述文件进行“分支覆盖”的相应操作,可以满足应用的要求。

4 结束语

本文基于 XML 技术提出一种扩展性更好的系统框架设计方案。该设计方案简化了软件测试工具系统框架中间结构部分的设计与实现,使系统对于应用的增加具有易于扩展的特性。通过本文实现的原型系统,验证了设计方案的有效性。

参考文献:

[1] 李茜,梅琳,凌辉,等. EASTT:一种嵌入式应用软件测试系统[J]. 计算机工程与科学,2002,24(2):66-69.

[2] 杜晓东. 面向嵌入式系统的测试工具研究[D]. 成都:电子科技大学,2003.

[3] SUN Changai, LIU Chao, JIN Maozhong, et al. Architecture framework for software test tool[C]//Proceedings 36th International Conference on Technology of Object-Oriented Language and Systems, TOOLS-Asia2000, 2000:40-47.

[4] KUNG D C, HSIA P, TOYOSHIMA Y, et al. Object-oriented software testing-some research and development

[C] //Third IEEE International High-Assurance Systems Engineering Symposium, 1998: 158 - 165.
[5] 许晓春,梅琳,胥光辉,等. EASTT 源码解析系列之二——中间结构管理器[J]. 共创软件,2002(1):35 - 38.
[6] 刘久富,孙德敏,杨忠,等. 嵌入式软件的动态测试[J]. 微计算机信息,2006,22(2):82 - 84.
[7] 晏华,袁海东,尹立孟. 代码自动插装技术的研究与实现[J]. 电子科技大学学报,2002,31(1):62 - 66.
[8] 钟治平,徐拾义. 程序插装技术在软件内建自测试中的应用[J]. 计算机工程与应用,2004(17):117 - 118.
[9] 唐科,汪文勇,刘利枚. 嵌入式软件覆盖测试的研究[J]. 成都信息工程学院学报,2005,20(5):541 - 545.
[10] 文昌辞,王昭顺. 软件测试自动化静态分析研究[J]. 计算机工程与设计,2005,26(4):987 - 989.

Application of XML technology in software test framework

XU GuangHui[1]　DU Yu[1]　LI Xiang[2]　XU YongSen[3]
(1. Institute of Command Automation, PLA University of Science and Technology, Nanjing, Jiangsu 210007;
2. East China Testing Center of Engineering Software, Shanghai 200233;
3. Department of Computer Science and Technology, Nanjing University, Nanjing, Jiangsu 210093, China)

Abstract: The system framework of software test tool is analysed. Based on result of the analysis, a well-extended system framework is presented. In the presented system framework, XML file format is used for the Source Code Information File. Different Info-get Template File, which is in XSLT format, is used to get the application-related information. As the application extends, the system only need to extend the Info-get Template File without having to change the Source Code Information File.
Key words: software testing; system framework; XML; XSLT

(上接第 100 页)

Research and practice of component-based software testability model

FU Cheng[1]　GONG YunZhan[2]　HONG Hui[1]
(1. Institute of China Electronic Systems Equipment Company, Beijing 100039;
2. College of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: This paper analyses the development of an military component-based software. Based on others' work, it defines a pentagon-based measurement rule for component-based software. This method can reflect the testability factors of the component being tested correctly, make the developers realize the defaults of that component, and it can also help developers to design components with well testability. Through the authors' test case experience, the pentagon-based measurement rule is proved helpful to increase the testability of the component-based software, and being certificated by both develop engineers and test engineers.
Key words: software component; testability; software test